




PYTHON







Python
is the name
of a programming
language
as well

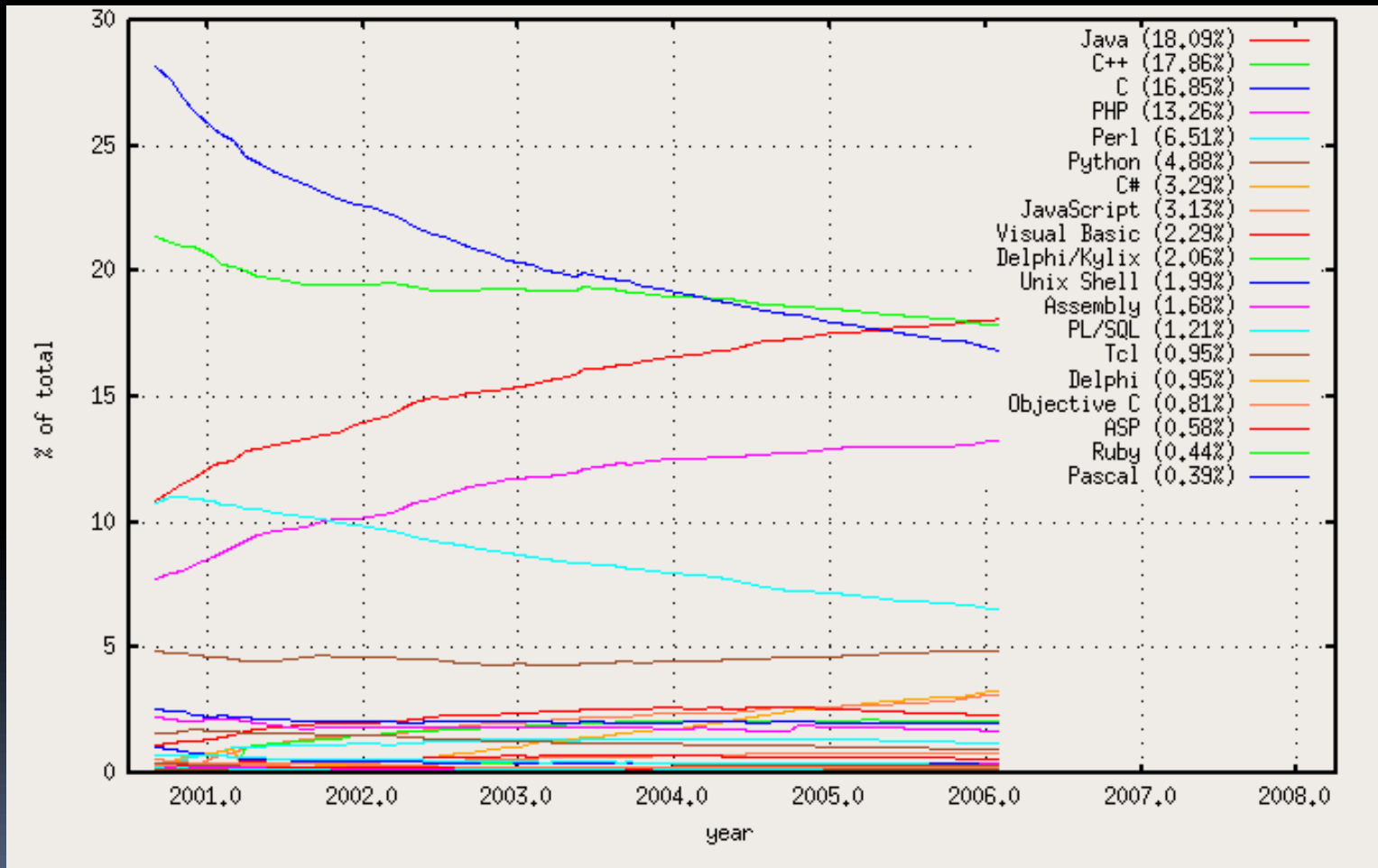
Programming Languages Usage

Position	Language	Ratings
1	Java	19.1%
2	C	15.2%
3	C++	10.1%
4	PHP	8.7%
5	Visual Basic	8.4%
6	Perl	6.2%
7	Python	3.8%
8	C#	3.7%
9	JavaScript	3.1%
10	Ruby	2.6%

The numbers are from May 2007.

www.zetcode.com/wxpython/introduction

Programming Languages Usage





Python

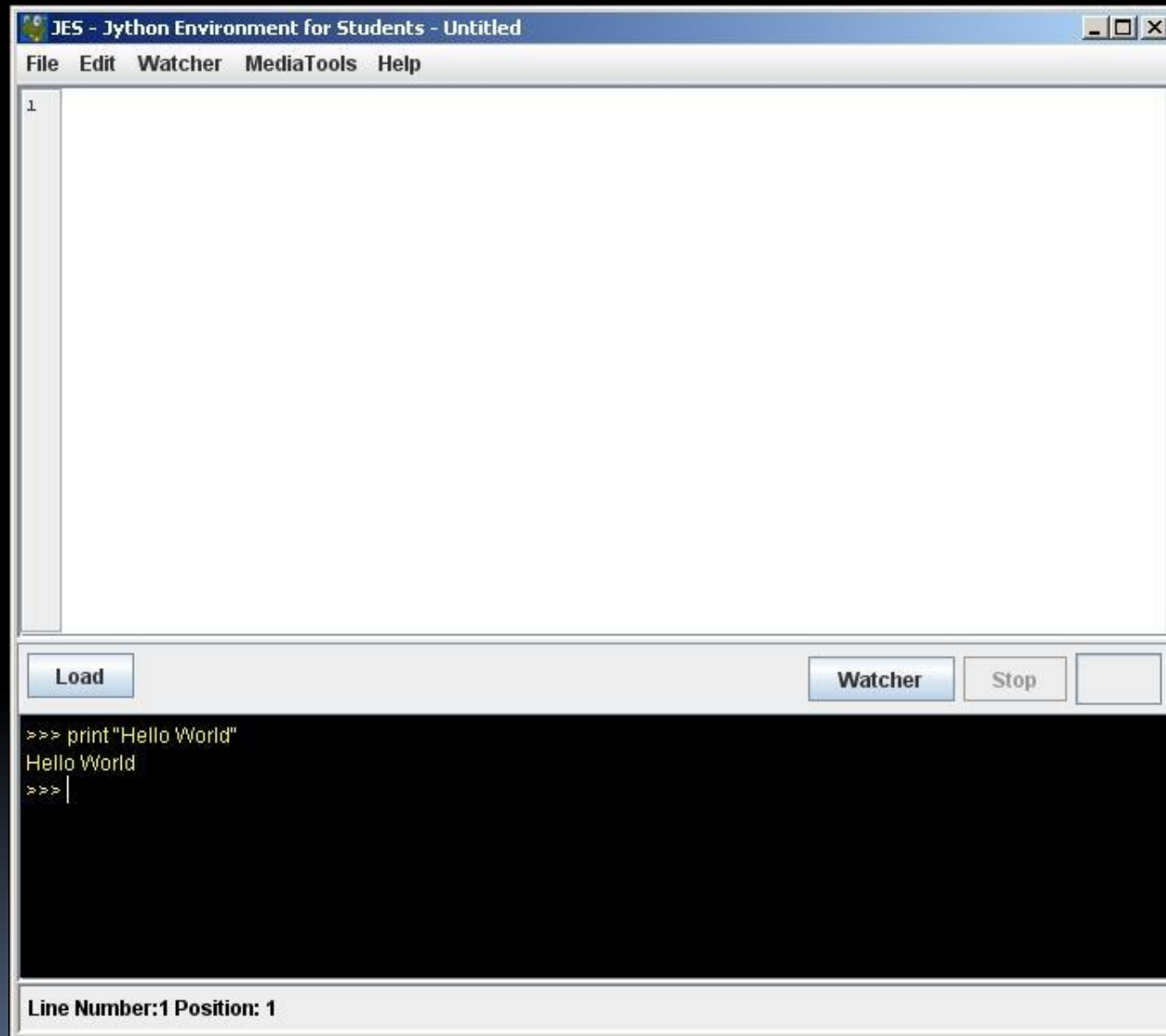
- Open Source (Free)
 - Portable
 - Simple Syntax
 - Well Documented
 - www.python.org
- 

Programming with Python

- Data
 - Constants and Variables
- Instructions (Code)
 - Assignment (=)
 - If .. Then statement
 - Loops

.....

JES



Variables

Variables are names for data

Examples of good variable names:

age = 99

province = "Alberta"

x = 15

y = 32

gasprice = 0.95

pi = 3.14

Variables (names)

Examples of incorrect variable names:

- `76trombones = "big parade"`

Not good because the variable name starts with a number .

- `more$ = 1000000`

Not good because the variable name has the dollar sign.

- `class = "Computer Science 203"`

Not good because the variable name used 'class' is a reserved word

Variables (Data Types)

- Numbers:

integers (0, 12, 17, -20,...),

floats (0.0, 3.14, 6.02e23,.....)

- Operators:

+ - * / % (modulus or remainder) ** (exponentiation)

- Strings:

“anything in double quotes”

Variables (Data Types)

- `Type("Hello World") : <type 'str'>`
- `type(10) : <type 'int'>`

Issues:

```
print 1,000,000 : 1 0 0
```


Variables (names)

Reserved words

and - assert - break - class - continue - def
del - elif - else - except - exec - finally
for - from - global - if - import - in
is - lambda - not - or - pass - print
raise - return - try - while - yield



Some practice

- Create a variable call it var1 and store the number 9 in it.
 - Create a second variable call it var2 and store the number 1 in it.
 - Get the sum of var1 and var2 and store the result in var3.
 - Switch the values of var1 and var2.
- 

Very Important

Python is case sensitive

Variable names:

- `X != x` (X is different from x)
- `VariableName != variableName`

Function names

- `Print != print`

if statement

```
if {conditions to be met}:  
    {do this}  
    {and this}  
    {and this}  
{but this happens regardless}  
{because it isn't indented}
```

Example 01

```
y = 1  
if y == 1:  
    print "y still equals 1, I was just checking"
```


Comparison operators

- == equal
- >= grater than or equal
- <= less than or equal
- < less than
- > grater than
- != different

if statement

Example02

```
a = 1
if a > 5:
    print "This shouldn't happen."
else:
    print 'This should happen.'
```

Example03

```
z = 4
if z > 70:
    print "Something is very wrong"
elif z < 7:
    print "This is normal"
```

Loops – for loop

for item **in** container:

action to repeat for each item in the container

else:

action to take once we have finished the loop.

Example01

```
sum = 0
```

```
for x in [1, 2, 3, 4]:
```

```
    sum = sum + x
```

```
execfile("forexample01")
```

Loops – for loop

Example02

```
newList = [45, 'eat me', 90210, 'The day has come', -67]  
for value in newList:  
    print value
```

```
execfile("forexample02")
```

Loops - while loop

while {condition that the loop continues}:

{what to do in the loop}

{have it indented, usually four spaces}

{the code here is not looped}

{because it isn't indented}

Example 01

```
x = 10
```

```
while x != 0:
```

```
    print x
```

```
    x = x - 1
```

```
    print "wow, we have counted x down, and now it equals", x  
print "And now the loop has ended."
```

Loops - while loop

Example02

```
print "We will show the even numbers up to 20"
```

```
n = 1
```

```
while n <= 20:
```

```
    if n % 2 == 0:
```

```
        print n
```

```
    n = n + 1
```

```
print "there, done."
```

```
execfile("whileexample02")
```

Important notes

A condition in an if statement or a while loop can be a simple one:

expression **comparison Operator** expression

if (age > 100):

if (x%2 == 0):

while((i + j) < (k * l - m)):

Or a mix of several simple conditions:

logical operator (condition₁) **logical operator** (condition₂)
.....**logical operator**(condition_n)

if (age>100 **AND** salary>100000):

if(color == "red" **OR** color=="pink"):

goodstanding = true

If (**NOT** goodstanding):

Operators

Comparison Operators

- ==
- >=
- <=
- <
- >
- !=

Logical Operators

NOT

AND

OR

The difference between for & while

```
sum = 0
```

```
x = 1
```

```
while x <= 4:
```

```
    sum = sum + x
```

```
    x = x + 1
```

```
print sum
```

```
sum = 0
```

```
for x in [1, 2, 3, 4]:
```

```
    sum = sum + x
```

```
print sum
```

functions

- **Built-in functions.**

`print` "something", somethingelse , ...

Manipulating pictures:

Get a picture from the internet - Google : [free pictures rockies](#)

`makePicture, show, getColor, SetColor,.....`

```
picture = makePicture("C:\\Documents and Settings\\aguerbas\\Desktop\\python.jpg")
```

```
show(picture)
```

User Defined Functions

function with no parameters and no returned value

```
def hello():  
    print "Hello World!"
```

When you call this function using the interactive console like this:

```
hello()
```

It will print Hello World!

Simple Argument

```
def withParameters ( txt ):  
    print txt+txt
```

When you call this function using the interactive console like this:

```
withParameters("Hello")
```

This function will print the Hello 2 times.

User Defined Functions

function with parameters and with returned value

```
def product (number1,number2):  
    result = number1 * number2  
    return result
```

The value returned by a function can be stored in a variable like this:

```
variableName = product(2,3)  
print variableName
```

The previous instructions can be in the same file where the function product is defined or can be typed in the interactive (black) console

Using Built-in function math functions

#example using built-in math functions

```
import math
```

```
def printLogarithm(x):
```

```
    if x <= 0:
```

```
        print "Positive numbers only, please."
```

```
        return
```

```
    result = math.log(x)
```

```
    print "The log of x is", result
```

Using Built-in function math functions

#example using built-in math functions

#using the return keyword

import math

def printLogarithm(x):

if x <= 0:

print "Positive numbers only, please."

return

return math.log(x)

Operations on Strings

```
message = "Good Morning"
```

The variable message has type String:

- The following operations are illegal:

```
message - 1
```

```
message * "Hello"
```

- Also the following operations are illegal:

```
"Hello" / 123
```

```
"15" + 2
```

Operations on Strings

- The following operations are legal:

```
Fruit = "banana"
```

```
bakedGood = "nut bread"
```

```
print fruit + bakedGood
```

- Repetition:

```
print "fun"*3    it gives you: "funfunfun"
```


Operations on numbers

- $2**1+1$ is 3 and not 4
- $3*1**3$ is 3 and not 27
- $2*3-1$ is 5 and not 4
- $2/3-1$ is -1 and not 1

(integer division: use decimals to get real numbers $2.0/3$)

- To avoid ambiguity (confusion) use parenthesis:

- $(2**1)+1$
- $3*(1**3)$
- $(2*3)-1$

Things to remember

- Python is case sensitive
- Indentation is very important in JES
- **Steps to run a Jython (JES) program**
 - Write your code (instructions)
 - Load the code (you will be asked to save it)
 - Run the code either by using
 - `execfile("location & name of the file")` command or by calling directly a function if you have one.
 - Instructions are executed sequentially



Exercise

- Write a piece of code that gives you the number of days you have lived.

Include your name and a description of what the code does at the beginning of your python program.







Exercise

- Write a program that converts temperature in Celsius to Fahrenheit
 - 1) Take the temperature in Celsius and multiply 1.8.
 - 2) Add 32 degrees. The result is degrees Fahrenheit.

Exercise

- Write a program that takes four different grades as input and gives you a letter grade.
- Ex:
quiz01: 90
quiz02: 95
quiz03 : 93
quiz04: 91
The output should be: A

```
def letterGrade(g1,g2,g3,g4):
    finalG = (g1 + g2 + g3 + g4)/4.0
    if finalG > 95:
        lgrade = 'A+'
    elif finalG > 90:
        lgrade = 'A'
    elif finalG > 85:
        lgrade = 'B+'
    elif finalG > 80:
        lgrade = 'B'
    elif finalG > 75:
        lgrade = 'B-'
    elif finalG > 70:
        lgrade = "C"
    elif finalG > 65:
        lgrade = "C-"
    elif finalG > 60:
        lgrade = "D+"
    elif finalG > 55:
        lgrade = "D"
    elif finalG > 50:
        lgrade = "D-"
    else:
        lgrade = "F"
    return finalG,lgrade
```

- 
- 
- Now try to use two functions instead of one to answer the same question.
 - The first function will have as input (four parameters) four grades. It will compute the average grade and passes that average as a parameter to the second function. The second function will take that average as input (parameter), do some work and returns the letter grade corresponding to that average grade.
 - Now the first function has the average grade(computed by itself) and the letter grade (computed by the second function). The only thing that it needs to do is printing both of them.
 - See the answer in the following slide.

#--- Beginning of letterGrade() function

```
def letterGrade(g):
```



```
    if g > 95:
        lg = 'A+'
    elif g > 90:
        lg = 'A'
    elif g > 85:
        lg = 'B+'
    elif g > 80:
        lg = 'B'
    elif g > 75:
        lg = 'B-'
    elif g > 70:
        lg = "C"
    elif g > 65:
        lg = "C-"
    elif g > 60:
        lg = "D+"
    elif g > 55:
        lg = "D"
    elif g > 50:
        lg = "D-"
    else:
        lg = "F"
    return lg
```

#--- End of letterGrade() function

#--- Beginning of letterGradeMain() function

```
def letterGradeMain(g1,g2,g3,g4):
    finalG = (g1 + g2 + g3 + g4)/4.0
    lgrade = letterGrade(finalG)
    return finalG,lgrade
```

#--- End of letterGradeMain() function

- 
- 
- Add a third function that computes the average grade.
 - The main function will pass a list of grades to a function that will compute the average grade only and return it to the main function.
 - Now, the main function has the average grade computed by someone else. The main function will call someone else also (another function) to find the letter grade that corresponds to that average grade.
 - Once the main function gets the letter grade from that other function it will just print the average grade and the letter grade corresponding to it.
 - See the answer in the following slide.

```
def letterGrade(g):
```

```
    if g > 95:
        lg = 'A+'
    elif g > 90:
        lg = 'A'
    elif g > 85:
        lg = 'B+'
    elif g > 80:
        lg = 'B'
    elif g > 75:
        lg = 'B-'
    elif g > 70:
        lg = 'C'
    elif g > 65:
        lg = 'C-'
    elif g > 60:
        lg = 'D+'
    elif g > 55:
        lg = 'D'
    elif g > 50:
        lg = 'D-'
    else:
        lg = 'F'
    return lg
```

```
#--- End of letterGrade() function
```

```
#--- Beginning of finalGrade() function
```

```
def finalGrade(allGrades):
```

```
    finalG = 0
    for i in allGrades:
        finalG = finalG + i
    finalG = finalG/len(allGrades)
    return finalG
```


```
#--- End of finalGrade() function
```

```
#--- Beginning of letterGradeMain() function
```

```
def letterGradeMain(g1,g2,g3,g4):
```

```
    grades = [g1, g2, g3, g4]
    fg = finalGrade(grades)
    lgrade = letterGrade(fg)
    return fg,lgrade
```

```
#--- End of letterGradeMain() function
```





```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sept", "Oct", "Nov", "Dec"]  
numberOfdays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

Write a function that prints the number of days of each month

1 – using a for loop

2- using a while loop





```
def months():
```

```
    months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sept", "Oct", "Nov", "Dec"]
```

```
    numberOfdays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

```
    if len(months) == len(numberOfdays):
```

```
        counter = 0
```

```
        while counter < len(months):
```

```
            print months[counter], " has ", numberOfdays[counter], " days"
```

```
            counter=counter+1
```

```
    else:
```

```
        print "both lists must have the same length"
```



def months():

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sept", "Oct", "Nov", "Dec"]
```

```
numberOfdays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```


```
if len(months) == len(numberOfdays):
```


```
    for i in range(12):
```

```
        print months[i], " has ", numberOfdays[i], " days"
```

```
else:
```

```
    print "both lists must have the same length"
```






`numberOfdays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]`

Write a function that returns the number of days of the shortest month.





```
def daysofShortestmonth():
```

```
    numberOfdays = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

```
    shortestMonth = 0
```

```
    for index in range(12):
```

```
        if numberOfdays[index]<numberOfdays[shortestMonth]:
```

```
            shortestMonth = index
```

```
    print "The",shortestMonth+1,"month is the shortest and its number of days is",numberOfdays[shortestMonth]
```